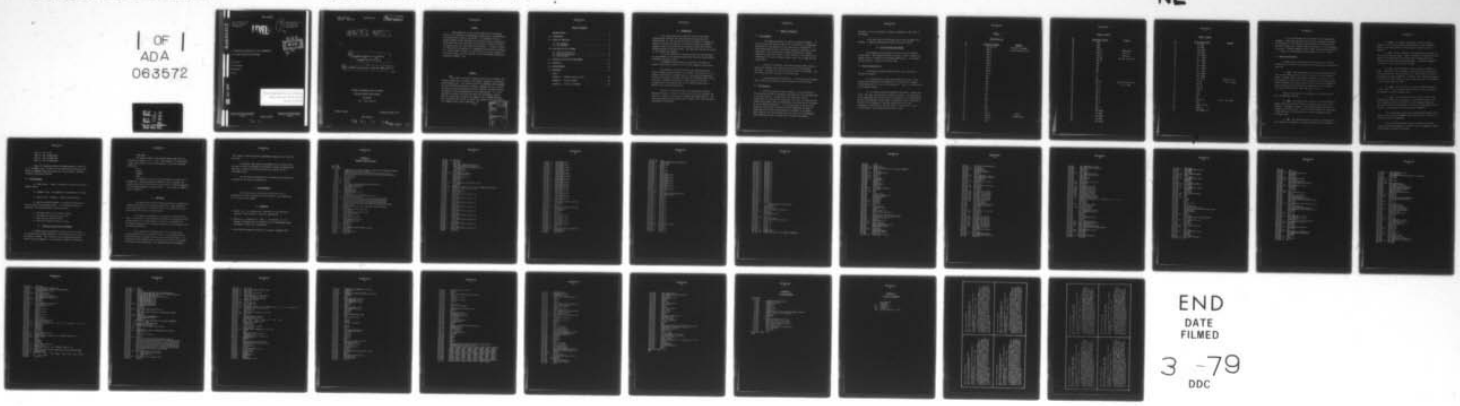
AD-A063 572 DEFENCE RESEARCH ESTABLISHMENT VALCARTIER (QUEBEC) F/G 9/2
AN IMPROVED VERSION OF AN ASSEMBLER AND SIMULATOR FOR THE 8080,(U)
NOV 78 J N BERUBE, R CARBONNEAU, B MONTMINY

UNCLASSIFIED DREV-R-4124/78 NL

| OF |
ADA
063572



END
DATE
FILMED
3 -79
DDC

UNCLASSIFIED

CRDV RAPPORT 4124/78
DOSSIER: 3633A-010
NOVEMBRE 1978

DREV REPORT 4124/78
FILE: 3633A-010
NOVEMBER 1978

AN IMPROVED VERSION OF AN ASSEMBLER AND SIMULATOR FOR THE 8080

J.N. Bérubé

R. Carbonneau

B. Montminy

P. Côté

Centre de Recherches pour la Défense
Defence Research Establishment
Valcartier, Québec

BUREAU - RECHERCHE ET DEVELOPPEMENT
MINISTERE DE LA DÉFENSE NATIONALE
CANADA

RESEARCH AND DEVELOPMENT BRANCH
DEPARTMENT OF NATIONAL DEFENCE
CANADA

NON CLASSIFIÉ

CRDV R-4124/78
DOSSIER: 3633A-010

UNCLASSIFIED

(14) DREV-R-4124/78
FILE: 3633A-010

(11) Nov 78 (12) 37 p.

(6) AN IMPROVED VERSION OF AN ASSEMBLER AND SIMULATOR FOR THE 8080

by

(10) J.N. Bérubé, R. Carbonneau, B. Montminy ~~and~~ P. Côté

CENTRE DE RECHERCHES POUR LA DEFENSE

DEFENCE RESEARCH ESTABLISHMENT

VALCARTIER

Tel: (418) 844-4271

Québec, Canada

November/novembre 1978

NON CLASSIFIE

RESUME

Nous présentons ici une version améliorée d'un programme servant à la traduction des mnémoniques employés dans la programmation du microprocesseur 8080 en code machine. Ce programme, écrit en langage FORTRAN, permet également la simulation exacte du comportement du microprocesseur dans des applications réelles de même que l'emploi d'étiquettes pour les instructions de branchement. Il permet en outre la simulation d'interruptions et l'impression des résultats intermédiaires. Cette nouvelle version, en plus d'être dix fois plus rapide que la version en APL, peut être utilisée sur tout ordinateur possédant un compilateur FORTRAN. (NC)

ABSTRACT

This report describes an improved version of an assembler and simulator being used to translate man-readable statements into machine-understandable code. This program, written in FORTRAN, allows programming of the 8080 microprocessor in symbolic language as well as the use of labels for jump instructions while the simulator duplicates exactly the behavior of the microprocessor in real applications. It is also possible to simulate interrupts and print out intermediate results. This new version may be used on any computer with a FORTRAN compiler while the former APL version was slower by a factor of ten and limited to a computer with an APL interpreter. (U)

TABLE OF CONTENTS

1.0 INTRODUCTION

This report describes an improved version of the 8080 microprocessor assembler and simulator previously discussed and published (Ref 1). The new version written in FORTRAN-IV language (as opposed to APL before) makes use of files for storing any assembler program as well as the simulated microprocessor memory which permit program editing of unlimited length. The former APL version was limited to 1500 instruction programs.

It is also possible to transfer directly the data generated by the assembler to the erasable programmable read-only memory (EPROM) programmer described in Ref 2.

The FORTRAN language permits a much faster execution of program instructions as well as the use of the computer system editor.

The assembler will first translate the 8080 program into machine language, give an address to all labels, find any error in the program, and execute the program exactly as would the microprocessor. It will also simulate interrupts and print out intermediate data during execution. Although this program is usable only with the 8080 microprocessor, the technique used in the FORTRAN routines may be applied to any other microprocessors.

Section 2.0 briefly describes the assembler and simulator operating principle. Section 3.0 details the use of the assembler and simulator. Section 4.0 outlines the EPROM programming procedure. The work was performed at DREV between May and August, 1976 under PCN 33A10, "Improvement to Equipment".

2.0 THEORY OF OPERATION

2.1 The assembler

The program assembling is done by a two-pass assembler. In the first pass, the binary coding of all instructions is accomplished except that of jump addresses in all jump and call instructions. A table containing all labels, with their respective addresses, is also constructed. Any syntax errors, illegal instructions or illegal arguments will also be detected in the first pass. The second pass is exclusively used to give values to bytes 2 and 3 of all jump and call instructions.

During the assembling, the read-only memory (ROM) content is kept on the computer file system up to a maximum of 65536 memory positions. Two other files are also built during the assembling; they are scratch pad files used only in the simulation.

The present program limitations are 5000 lines and 500 labels. These limits are artificial and may be changed if the need exists.

2.2 The simulator

After the program has been assembled, it may be simulated. The simulation faithfully reproduces the operation of the microprocessor in real applications. In particular, the jumps, the calls to subroutines, the returns from subroutines, the stack, the flags and the random access memory (RAM) are manipulated in the same way as they would be in the 8080 microprocessor. It is also possible to print out the processor status at any time with the PRIN command and to output a memory map of 256 bytes located between 0 and 65535 in the memory. These printouts will not interfere with the result of the assembling. These commands are used to analyze and to monitor the program.

Similarly, it is also possible to simulate interrupts at any point in the program.

A few other special instructions may be used throughout the program. A complete description is given in the next section.

3.0 UTILIZATION OF THE PROGRAM

The program is first written in a file using the computer system editor. The program must then be copied in order to remove any deleted line or to reorder any fractional numbered line. The command is COPY X OVER X, 1 where X is the file name.

3.1 Basic instruction set

The following remarks should be taken into account while writing the program:

a) There must be at least one instruction per line. Each instruction may optionally be followed by a semicolon and a comment. The maximum length of a line is 128 characters. There is no restriction on the comment content.

b) An instruction may be optionally preceded by a label and a colon. The colon is used only if a label is present. The label may be of any length but only the first four (4) characters will be meaningful. During the assembling, a label table is constructed and is outputted at the end of this operation. The only restriction on the label characters is that they must not be a space, a semicolon or a colon. A nonexecutable instruction should not have a label.

TABLE I

Instruction set

No	Instruction format	Remarks
1	MOV R1, R2	R1=R2=M is illegal
2	INR R	R=A,B,C,D,E,H,L, or M
3	DCR R	
4	ADD R	
5	ADC R	
6	SUB R	
7	SBB R	
8	ANA R	
9	XRA R	
10	ORA R	
11	CMP R	
12	RLC	
13	RRC	
14	RAL	
15	RAR	
16	HLT	
17	RET	
18	RC	
19	RNC	
20	RZ	
21	RNZ	
22	RP	
23	RM	
24	RPE	
25	RPO	
26	RST A	$0 \leq A \leq 7$
27	PUSH RR	RP=PSW,B,D,H
28	POP RR	
29	XCHG	

TABLE I (contd)

No	Instruction format	Remarks
30	XTHL	
31	SPHL	
32	PCHL	
33	DAD RR	RR=SP,B,D,H
34	STAX R	R=B or D
35	LDAX R	R=B or D
36	INX RR	RR= SP, B, D or H
37	DCX RR	
38	CMA	
39	STC	
40	CMC	
41	DAA	
42	EI	
43	DI	
44	NOP	
45	MVI R,A	R=A,B,C,D,E,H,L or M
46	ADI A	$0 \leq A < 256$
47	ACI A	
48	SUI A	
49	SBI A	
50	ANI A	
51	XRI A	
52	ORI A	
53	CPI A	
54	IN A	
55	OUT A	
56	JMP LABEL	
57	JC LABEL	
58	JNC LABEL	
59	JZ LABEL	
60	JNZ LABEL	

TABLE I (contd)

No	Instruction format	Remarks
61	JP LABEL	
62	JM LABEL	
63	JPE LABEL	
64	JPO LABEL	
65	CALL LABEL	
66	CC LABEL	
67	CNC LABEL	
68	CZ LABEL	
69	CNZ LABEL	
70	CP LABEL	
71	CM LABEL	
72	CPE LABEL	
73	CPO LABEL	
74	LXI RR,AA	RR=B,D,H or SP
75	STA AA	$0 \leq$ AA < 65536
76	LDA AA	
77	SHLD AA	
78	LHLD AA	
79	INTE	
80	PRIN	
81	END	
82	ASSI AA,A	$0 \leq$ AA < 256 (data)
83	MAP A	
84	LABEL: VAR AA	
85	BASE AA	
86	ASSD LABEL, AA	

c) All instructions must be followed by their appropriate arguments. These are described in Ref 3 and are reproduced in Table 1 for convenience. There must be a space between the instruction and the first argument and a comma between the arguments of a two-argument instruction. Numbered argument N must respect the following convention:

N shall be a nonfractional decimal number with $0 \leq N < 65536$.

3.2 Special instructions

Several special instructions have been added to the basic instruction set in order to facilitate the assembling and the simulation. These are:

a) PRIN: This instruction is used without argument and will print out the processor status, the line counter, and the registers, namely: program counter, accumulator, registers, B, C, D, E, H, L, stack pointer, 5 flags (carry, zero, sign, parity and auxiliary carry), enable interrupt and a special counter which calculates the number of steps (clock pulses) elapsed since the beginning of the simulation (if the processor works at 2 MHz, one step is 0.5 μs).

PRIN will not affect the assembling and should be used to help in debugging a program.

b) INTE: This instruction simulates an interrupt that will happen exactly at the position where the instruction is put in the program. No argument is needed. The assembled program will not be changed by this instruction. This may be used anywhere in the program to know what will be the effect of an interrupt happening at this particular point.

c) END: This instruction must be the last instruction of a given program. No argument is needed and assembling is unaffected.

d) ASSI: This special instruction is used to assign a number ($0 \leq N < 256$) to a particular ROM address. One instruction should be used for each byte to be assigned. Simulation is not affected.

e) MAP: During simulation, this instruction will print out the content of a memory page. A page is formed by 256 consecutive bytes. Anyone of the 65536 bytes of the memory may be printed with this instruction without affecting the assembling. A numerical argument N is needed ($0 \leq N < 255$) to select the page.

f) VAR: The VAR command is used to assign an address to a label. However only the assembling is affected by this instruction. It should be used to assign an address to a label which is not part of the program now under assembling. Therefore, this instruction requires a label and the right argument of this instruction (VAR) is the address to be associated with the label.

g) BASE: This instruction is used to change the program counter value. The instruction following BASE will have the value of the argument of BASE as program counter.

h) ASSD: This instruction may be used to facilitate the addressing of a portion of a program in a relocatable context. The first argument must be a label and the second argument represents the address where the label address will be stored (two bytes are then written). This instruction affects only the assembling.

All nonexecutable instructions immediately following a CALL, JUMP or PCHL will not be executed during the simulation. There is no restriction for the assembling.

Once the FORTRAN program (given in Appendix A) has been compiled (the object program is called OBJ) and your program is ready, the procedure is started by entering

```
SET F:1,  DC/  X;IN
SET F:2,  DC/ F2;INOUT;SAVE
SET F:3,  DC/ F3;INOUT;SAVE
SET F:4,  DC/ F4;INOUT;SAVE
```

Where "X" identifies the file the 8080 program is stored in. F2 is the memory file. F3 and F4 are two additional scratch pad files. These set commands should be entered only once per LOGON. Execution of OBJ is initiated by RUN OBJ.

3.3 Error messages

a) Label error: A label is missing or there are two or more similar labels.

b) Argument error: The argument is unacceptable or missing.

c) Syntax error: Probably a nonexisting instruction.

d) BAD KEY or MISSING RECORD: An attempt has been made to write or read in an undefined record. If this happens during the simulation, the following things should be checked:

1- The stack pointer or the stack is wrong.
2- The program is longer than 5000 lines.
3- There are more than 500 labels.
4- The H and L register content is wrong.

4.0 INTERFACE WITH THE PROM PROGRAMMER

A FORTRAN program called MOR has been written to interface the assembler with a PROM programmer. This program should be compiled and linked to APLFNS. LPR to obtain an object program called MORP which is given in Appendix B. The execution of MORP is initiated by

START MORP

The question "What is the starting memory page?" must be answered by a number A ($0 \leq A \leq 252$). The program will then transfer 4 pages (1024 bytes) to an APL file named FA. After the following operations:

```
APL
)LOAD W
ASSPROM
)SAVE
```

are completed, a vector A of 1024 elements is saved in the workspace W. The main purpose of ASSPROM is thus to transfer the content of the APL file FA to the vector A which is directly compatible with the PROM programmer described in Ref 2. The listing of ASSPROM is given in Appendix C.

5.0 CONCLUSION

The improved version of the 8080 microprocessor assembler and simulator described in this report has permitted a speed increase in program execution time by a factor of 10 over the former APL version while eliminating most of its shortcomings.

The program is assembled in two passes and simulated afterwards. The simulator, which faithfully reproduces the functions of the microprocessor in real applications, will reduce considerably the writing and debugging time of programs employed in system designs using the 8080 microprocessor.

This assembler and simulator proved to be effective for developing 8080-based microcomputer systems. For many applications, where microcomputer systems are used to replace hardwire logic, it is essential for the program writer to keep a total control on the nature and timing of each instruction. Our assembler gives the programmer

this control, whereas high-level programming languages like the PL/M 80 do not.

Furthermore, many project developments may be carried out at the same time with the assembler and simulator without interfering with each other, otherwise each project would have required its own development system.

The technique developped for this assembler and simulator can be adapted to any type of microprocessor.

6.0 ACKNOWLEDGMENTS

The authors wish to express their gratitude to Mr. A. Blanchard for valuable comments and ideas brought to this program and to Dr. Giroux for his support.

7.0 REFERENCES

1. Bérubé, J.N., "An Assembler and a Simulator for the 8080 Microprocessor", DREV M-2402/76, June 1976, UNCLASSIFIED

2. Montminy, B., Carbonneau, R., Côté, P., and Bérubé, J.N., "An Automatic Programmer for the 2708/2704 Erasable Programmable Read Only Memory", DREV R-4131, UNCLASSIFIED

3. Intel 8080 Microcomputer Systems User's Manual, September 1975.

APPENDIX A

FORTRAN Program Listing

```
 EDIT LEC
EDIT HERE
*TY 1-1000
   1.000        DIMENSION I(32),IR12(4),LABR(500),IRR(10),N(3),IPCR(500),KS(256)
   2.000        COMMON IBUF(3,256),NC(3),IFL(3)
   3.000        DATA IEI,N,LABC,IPC,JJ,INS,IST,IL/0,0,44,55,0,0,0,0,0,1/
   4.000        DEFINE FILE 2(256,256,U,ICA),3(20,256,U,ICB),4(64,256,U,ICB)
   5.000        DO 476 J=1,3
   6.000        DO 486 K=1,256
   7.000 486    IBUF(J,K)=0
   8.000        NC(J)=1
   9.000 476    IFL(J)=0
  10.000 409    JJ=JJ+1
  11.000        CALL SI(JJ,0,LABP,LAB,INST,IR1P,IR2P,IR12,I)
  12.000        CALL REWR(JJ-1,2,IPC,1)
  13.000        IF(LABP.EQ.0)GOTO 400
  14.000        LABC=LABC+1
  15.000        LABR(LABC)=LAB
  16.000        IPCR(LABC)=IPC
  17.000 400    IF(INST.GE.79)GO TO 475
  18.000        DO 405 J=1,3
  19.000        IF(INST.GT.N(J)) CALL REWR(IL-1,3,JJ,1)
  20.000   405  IF(INST.GT.N(J))IL=IL+1
  21.000 475    GOTO(1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,
  22.000       120,21,22,23,24,25,26,27,28,29,30,31,32,33,34,35,36,
  23.000       237,38,39,40,41,42,43,44,45,46,47,48,49,50,51,52,53,
  24.000       354,55,56,57,58,59,60,61,62,63,64,65,66,67,68,69,70,
  25.000       471,72,73,74,75,76,77,78,409,409,81,82,409,84,85,409),INST
  26.000        PRINT 481,JJ
  27.000   481  FORMAT('AT LINE NO ',I5,' THE INST. DOES NOT EXIST')
  27.100        GOTO 409
  28.000    45  CALL REG(I,IR12(1),IR12(2),K)
  29.000        K=6+8*(K-1)
  30.000        CALL CTE(I,IR12(3),IR12(4),K1)
  31.000        CALL WRI(K,K1,IPC)
  32.000        GO TO 409
  33.000 245    CALL REWR(IPC,1,K,0)
  34.000        K=K/8
  35.000        CALL REWR(IPC+1,1,IX,0)
  36.000        IF(K.EQ.6)GO TO 484
  37.000        IRR(K+1)=IX
  38.000        IST=IST+7
  39.000 483    IPC=IPC+2
  40.000        GO TO 410
  41.000 484    CALL REWR(256*IRR(5)+IRR(6),1,IX,1)
  42.000        IST=IST+10
  43.000        GO TO 483
  44.000    44  CALL WR1(0,IPC)
  45.000         GOTO 409
```

```
46.000    244 IST=IST+4
47.000        IPC=IPC+1
48.000        GO TO 410
49.000 1      CALL REG (I,IR12(1),IR12(2),K)
50.000        CALL REG(I,IR12(3),IR12(4),K1)
51.000        IF(K1.EQ.7.AND.K.EQ.7)PRINT 414,JJ
52.000         K=55+K1+8*K
53.000         CALL WR1(K,IPC)
54.000        GO TO 409
55.000 201    CALL REWR(IPC,1,K1,0)
56.000        CALL REGR(IPC,IRR,K)
57.000        K1=K1/8¬8
58.000        IF(K1.EQ.6)GO TO 411
59.000        IRR(K1+1)=K
60.000        IST=IST+1
61.000        GO TO 244
62.000 411    CALL REWR(IRR(5)*256+IRR(6),1,K,1)
63.000        IST=IST+3
64.000        GO TO 244
65.000 414    FORMAT('AT LINE NO ',I5,' THE ARGUMENTS ARE ILLEGAL')
66.000 2      CALL REG(I,IR12(1),IR12(2),K)
67.000        K=8*K¬4
68.000 417    CALL WR1(K,IPC)
69.000        GO TO 409
70.000 3      CALL REG(I,IR12(1),IR12(2),K)
71.000        K=8*K¬3
72.000        GO TO 417
73.000 4      CALL REG(I,IR12(1),IR12(2),K)
74.000        K=127+K
75.000        GO TO 417
76.000 5      CALL REG(I,IR12(1),IR12(2),K)
77.000        K=135+K
78.000        GO TO 417
79.000 6      CALL REG(I,IR12(1),IR12(2),K)
80.000        K=143+K
81.000        GO TO 417
82.000 7      CALL REG(I,IR12(1),IR12(2),K)
83.000        K=151+K
84.000        GO TO 417
85.000 8      CALL REG(I,IR12(1),IR12(2),K)
86.000        K=159+K
87.000        GO TO 417
88.000 9       CALL REG(I,IR12(1),IR12(2),K)
89.000        K=167+K
90.000        GO TO 417
91.000 10     CALL REG(I,IR12(1),IR12(2),K)
92.000        K=175+K
93.000        GO TO 417
94.000 11     CALL  REG(I,IR12(1),IR12(2),K)
95.000        K=183+K
96.000        GO TO 417
```

```
 97.000 12     CALL WR1(7,IPC)
 98.000        GO TO 409
 99.000 13     CALL WR1(15,IPC)
100.000        GO TO 409
101.000 14      CALL WR1(23,IPC)
102.000        GO TO 409
103.000 15     CALL WR1(31,IPC)
104.000         GO TO 409
105.000 16     CALL WR1(118,IPC)
106.000        GO TO 409
107.000 17      CALL WR1(201,IPC)
108.000        GO TO 409
109.000 18     CALL WR1(216,IPC)
110.000        GO TO 409
111.000 19     CALL WR1(208,IPC)
112.000        GO TO 409
113.000 20     CALL WR1(200,IPC)
114.000        GO TO 409
115.000 21     CALL WR1(192,IPC)
116.000        GO TO 409
117.000 22     CALL WR1(240,IPC)
118.000        GO TO 409
119.000 23     CALL WR1(248,IPC)
120.000         GO TO 409
121.000 24     CALL WR1(232,IPC)
122.000        GO TO 409
123.000 25     CALL WR1(224,IPC)
124.000        GO TO 409
125.000 26     CALL CTE(I,IR12(1),IR12(2),K)
126.000        K=199+8*K
127.000        GO TO 417
128.000 27     K1=197
129.000 418    CALL REG(I,IR12(1),IR12(2),K)
130.000        IF(K.EQ.10) K=7
131.000        K=K1+8*(K-1)
132.000        GO TO 417
133.000 28     K1=193
134.000        GO TO 418
135.000 29     CALL WR1(235,IPC)
136.000        GO TO 409
137.000 30     CALL WR1(227,IPC)
138.000        GO TO 409
139.000 31     CALL WR1(249,IPC)
140.000        GO TO 409
141.000 32     CALL WR1(233,IPC)
142.000        GO TO 409
143.000 33     K1=9
144.000 420    CALL REG(I,IR12(1),IR12(2),K)
145.000        IF(K.EQ.9) K=7
146.000        K=K1+8*(K-1)
147.000        GO TO 417
```

```
148.000 34     K1=2
149.000 419     CALL  REG(I,IR12(1),IR12(2),K)
150.000        K=K1+8*(K-1)
151.000        GO TO 417
152.000 35     K1=10
153.000        GO TO 419
154.000 36     K1=3
155.000        GO TO 420
156.000 37     K1=11
157.000        GO TO 420
158.000 38     CALL WR1(47,IPC)
159.000        GO TO 409
160.000 39     CALL WR1(55,IPC)
161.000        GO  TO 409
162.000 40     CALL WR1(63,IPC)
163.000        GO TO 409
164.000 41     CALL WR1(39,IPC)
165.000        GO TO 409
166.000  42    CALL WR1(251,IPC)
167.000        GO TO 409
168.000 43     CALL WR1(243,IPC)
169.000        GO TO 409
170.000 46     K1=198
171.000 421    CALL CTE(I,IR12(1),IR12(2),K)
172.000        CALL WRI(K1,K,IPC)
173.000        GO TO 409
174.000 47     K1=206
175.000        GO TO 421
176.000 48     K1=214
177.000        GO TO 421
178.000 49     K1=222
179.000        GO TO 421
180.000 50     K1=230
181.000        GO TO 421
182.000 51     K1=238
183.000         GO TO 421
184.000 52     K1=246
185.000        GO TO 421
186.000 53     K1=254
187.000        GO TO 421
188.000 54     K1=219
189.000         GO TO 421
190.000 55     K1=211
191.000        GO TO 421
192.000 56     K1=195
193.000 422    CALL WRI(K1,0,IPC)
194.000        GO TO 44
195.000 57     K1=218
196.000        GO TO 422
197.000 58     K1=210
198.000        GO TO 422
```

```
199.000 59      K1=202
200.000         GO TO 422
201.000 60      K1=194
202.000         GO TO 422
203.000 61      K1=242
204.000         GO TO 422
205.000 62      K1=250
206.000         GO TO 422
207.000 63      K1=234
208.000         GO TO 422
209.000 64      K1=226
210.000         GO TO 422
211.000 65      K1=205
212.000         GO TO 422
213.000 66      K1=220
214.000         GO TO 422
215.000 67      K1=212
216.000         GO TO 422
217.000 68      K1=204
218.000         GO TO 422
219.000 69      K1=196
220.000         GO TO 422
221.000 70      K1=244
222.000         GO TO 422
223.000 71      K1=252
224.000         GO TO 422
225.000 72      K1=236
226.000         GO TO 422
227.000 73      K1=228
228.000         GO TO 422
229.000 74          CALL REG(I,IR12(1),IR12(2),K)
230.000         IF(K.EQ.9)K=7
231.000         K=1+8*(K-1)
232.000         CALL WR1(K,IPC)
233.000         CALL CTE(I,IR12(3),IR12(4),K)
234.000 423     K1=K/256
235.000         K=K-K1*256
236.000         CALL WRI(K,K1,IPC)
237.000         GO TO 409
238.000 75      K1=50
239.000 424     CALL WR1(K1,IPC)
240.000         CALL CTE(I,IR12(1),IR12(2),K)
241.000         GO TO 423
242.000 76      K1=58
243.000         GO TO 424
244.000 77      K1=34
245.000         GO TO 424
246.000 78      K1=42
247.000         GO TO 424
262.000 242     PRINT 426,JJ
263.000 426     FORMAT('AT LINE NO',I5,'ENABLE INTERRUPT')
```

```
264.000         IEI=1
265.000         GO TO 244
266.000 243     PRINT 427,JJ
267.000 427     FORMAT('AT LINE NO',I5,'DISABLE INTERRUPT')
268.000         IEI=0
269.000         GO TO 244
270.000 240     ICAR=IABS(ICAR-1)
271.000         GO TO 244
272.000 239     ICAR=1
273.000         GO TO 244
274.000 229      K=IRR(3)
275.000         IRR(3)=IRR(5)
276.000         IRR(5)=K
277.000         K=IRR(4)
278.000         IRR(4)=IRR(6)
279.000         IRR(6)=K
280.000         GO TO 244
281.000 231     ISP=IRR(6)+256*IRR(5)
282.000         GO TO 434
283.000 238     K2=0
284.000         K1=1
285.000         DO 428 J=1,8
286.000         K=IRR(8)-(IRR(8)/2)*2
287.000         IRR(8)=IRR(8)/2
288.000         K=IABS(K-1)
289.000         K2=K2+K*K1
290.000 428     K1=K1+K1
291.000         IRR(8)=K2
292.000         GO TO 244
293.000  241    K=0
294.000          K1=IRR(8)/16
295.000         K2=IRR(8)-K1*16
296.000         K3=K2
297.000         IF(K2.GE.10) K=1
298.000         IF(K+ICY1.GE.1) K2=K2+6
299.000         K=0
300.000         IF((K3.GE.10).AND.(K1.EQ.9) ) K=1
301.000         IF((ICAR.EQ.1).OR.(K1.GE.10))K=K+1
302.000         IF(K.EQ.0) GO TO 429
303.000         ICAR=1
304.000 429     IRR(8)=K2+K1*16
305.000         GO TO 244
306.000 212     IRR(8)=IRR(8)*2
307.000         CALL CAR(ICAR,IRR(8),256)
308.000         IRR(8)=IRR(8)+ICAR
309.000         GO TO 244
310.000 213     K1=IRR(8)/2
311.000         ICAR=IRR(8)-K1*2
312.000         IRR(8)=K1+ICAR*128
313.000         GO TO 244
314.000 214     IRR(8)=IRR(8)*2+ICAR
```

```
315.000         CALL CAR(ICAR,IRR(8),256)
316.000         GO TO 244
317.000 215     K1=IRR(8)/2
318.000         K2=IRR(8)-K1*2
319.000         IRR(8)=K1+ICAR*128
320.000         ICAR=K2
321.000         GO TO 244
322.000 256     CALL REWR(IPC+1,1,K1,0)
323.000         CALL REWR(IPC+2,1,K2,0)
324.000         IPC=K1+K2*256
325.000         IST=IST+10
326.000 469     CALL REWR(IPC,3,K1,0)
327.000         JJ=K1-1
328.000         GO TO 410
329.000 257     IF(ICAR.EQ.1) GO TO 256
330.000 432     IPC=IPC+2
331.000         IST=IST+6
332.000         GO TO 244
333.000 258     IF (ICAR) 432,256,432
334.000 259     IF(IZER) 432,432,256
335.000 260     IF(IZER ) 432,256,432
336.000 261     IF(ISIG) 432,256,432
337.000 262     IF(ISIG) 432,432,256
338.000 263     IF(IPAR) 432,432,256
339.000 264     IF(IPAR) 432,256,432
340.000 265     K1=(IPC+3)/256
341.000         K=(IPC+3)-K1*256
342.000         CALL REWR(ISP-1,1,K1,1)
343.000         CALL REWR(ISP-2,1,K,1)
344.000         ISP=ISP-2
345.000         IST=IST+7
346.000         GO TO 256
347.000 266     IF(ICAR.EQ.1) GO TO 265
348.000 433     IPC=IPC+2
349.000         IST=IST+7
350.000         GO TO 244
351.000 267     IF(ICAR) 433,265,433
352.000 268     IF(IZER)433,433,265
353.000 269     IF(IZER)433,265,433
354.000 270     IF(ISIG) 433,265,433
355.000 271     IF(ISIG) 433,433,265
356.000 272     IF(IPAR) 433,433,265
357.000 273     IF(IPAR) 433,265,433
358.000 232     IPC=IRR(6)+256*IRR(5)
359.000         IST=IST+5
360.000         CALL REWR(IPC,3,K1,0)
361.000         JJ=K1-1
362.000         GO TO 410
363.000 230     CALL REWR(ISP,1,K1,0)
364.000         CALL REWR(ISP,1,IRR(6),1)
365.000         IRR(6)=K1
```

```
366.000          CALL REWR(ISP+1,1,K1,0)
367.000          CALL REWR(ISP+1,1,IRR(5),1)
368.000          IRR(5)=K1
369.000          IST=IST+14
370.000          GO TO 244
371.000 217      CALL REWR(ISP,1,K1,0)
372.000          CALL REWR(ISP+1,1,K2,0)
373.000          ISP=ISP+2
374.000          IPC=K2*256+K1
375.000          IST=IST+11
376.000          GO TO 469
377.000 218      IF(ICAR.EQ.1) GO TO 217
378.000 434      IST=IST+1
379.000          GO TO 244
380.000 219      IF(ICAR) 434,217,434
381.000 220      IF(IZER)434,434,217
382.000 221      IF(IZER)434,217,434
383.000 222      IF(ISIG) 434,217,434
384.000 223      IF(ISIG)434,434,217
385.000 224      IF(IPAR)434,434,217
386.000 225      IF(IPAR) 434,217,434
387.000 255      CALL REWR(IPC+1,1,K1,0)
388.000          PRINT 435,JJ,K1,IRR(8)
389.000 435      FORMAT('AT LINE NO ',I5,' OUTPUT NO ',I3,' = ',I3)
390.000          IPC=IPC+2
391.000          IST=IST+10
392.000          GO TO 410
393.000 226      CALL REWR(IPC,1,K1,0)
394.000          K3=((K1-192)/8)*8
395.000  501     K2=(IPC+1)/256
396.000          K=(IPC+1)-K2*256
397.000          CALL REWR(ISP-1,1,K2,1)
398.000          CALL REWR(ISP-2,1,K,1)
399.000          ISP=ISP-2
400.000          IPC=K3
401.000          IST=IST+11
402.000          CALL REWR(IPC,3,K1,0)
403.000          JJ=K1-1
404.000          GO TO 410
405.000 204      CALL REGR(IPC,IRR,K)
406.000 437      CALL ICY(ICY1,K,IRR(8))
407.000 439      IRR(8)=IRR(8)+K
408.000 447      CALL CAR(ICAR,IRR(8),256)
409.000          CALL FPZSP(IZER,ISIG,IPAR,IRR(8))
410.000          GO TO 244
411.000 246      CALL REWR(IPC+1,1,K,0)
412.000 438      IST=IST+3
413.000          IPC=IPC+1
414.000          GO TO 437
415.000 205      CALL REGR(IPC,IRR,K)
416.000          K=K+ICAR
```

```
417.000         GO TO 437
418.000 247     CALL REWR(IPC+1,1,K,0)
419.000         K=K+ICAR
420.000         GO TO 438
421.000 206     CALL REGR(IPC,IRR,K)
422.000 440     K=-K
423.000         GO TO 439
424.000 248     CALL REWR(IPC+1,1,K,0)
425.000         IST=IST+3
426.000         IPC=IPC+1
427.000         GO TO 440
428.000 207     CALL REGR(IPC,IRR,K)
429.000 441     K=-K-ICAR
430.000         GO TO 439
431.000 249     CALL REWR(IPC+1,1,K,0)
432.000         IST=IST+3
433.000         IPC=IPC+1
434.000         GO TO 441
435.000 208     CALL REGR(IPC,IRR,K)
436.000 443     K6=2
437.000         I1=2
438.000 444     K4=0
439.000         ICAR=0
440.000         ICY1=0
441.000         K5=1
442.000         DO 442 J=1,8
443.000         K3=0
444.000          K1=K-(K/2)*2
445.000         K2=IRR(8)-(IRR(8)/2)*2
446.000         IF((K1+K2).EQ.K6.OR.(K1+K2).EQ.I1)K3=1
447.000         K4=K3*K5+K4
448.000         IRR(8)=IRR(8)/2
449.000         K=K/2
450.000 442     K5=K5*2
451.000         IRR(8)=K4
452.000         CALL FFZSP(IZER,ISIG,IPAR,IRR(8))
453.000         GO TO 244
454.000 250     CALL REWR(IPC+1,1,K,0)
455.000         IST=IST+3
456.000         IPC=IPC+1
457.000         GO TO 443
458.000 209     CALL REGR(IPC,IRR,K)
459.000 445      K6=1
460.000         I1=1
461.000         GO TO 444
462.000 251     CALL REWR(IPC+1,1,K,0)
463.000         IST=IST+3
464.000         IPC=IPC+1
465.000         GO TO 445
466.000 210     CALL REGR(IPC,IRR,K)
467.000  446    K6=1
```

```
468.000         I1=2
469.000         GO TO 444
470.000 252     CALL REWR(IPC+1,1,K,0)
471.000         IST=IST+3
472.000         IPC=IPC+1
473.000         GO TO 446
474.000 211     CALL REGR(IPC,IRR,K)
475.000 448     K=IRR(8)-K
476.000         CALL CAR(ICAR,K,256)
477.000         CALL FFZSP(IZER,ISIG,IPAR,K)
478.000         GO TO 244
479.000 253     CALL REWR(IPC+1,1,K,0)
480.000         IST=IST+3
481.000         IPC=IPC+1
482.000          GO TO 448
483.000 202     K1=1
484.000 450     CALL REWR(IPC,1,K,0)
485.000         K=K/8
486.000         IF(K.EQ.6) GO TO 449
487.000         IRR(K+1)=IRR(K+1)+K1
488.000         CALL CAR(K3,IRR(K+1),256)
489.000         CALL FFZSP(IZER,ISIG,IPAR,IRR(K+1))
490.000         IST=IST+1
491.000         GO TO 244
492.000 449     CALL REWR(IRR(5)*256+IRR(6),1,K2,0)
493.000         K2=K2+K1
494.000         CALL CAR(K3,K2,256)
495.000         CALL FFZSP(IZER,ISIG,IPAR,K2)
496.000         CALL REWR(IRR(5)*256+IRR(6),1,K2,1)
497.000         IST=IST+5
498.000         GO TO 244
499.000 203     K1=-1
500.000         GO TO 450
501.000 275     CALL REWR(IPC+1,1,K1,0)
502.000         CALL REWR(IPC+2,1,K2,0)
503.000         K1=K2*256+K1
504.000         GO TO (375,276,277,278) INST-74
505.000 375     CALL REWR(K1,1,IRR(8),1)
506.000 451     IST=IST+13
507.000         IPC=IPC+3
508.000         GO TO 410
509.000 276     CALL REWR(K1,1,IRR(8),0)
510.000         GO TO 451
511.000 277     K2=1
512.000 452     CALL REWR(K1,1,IRR(6),K2)
513.000         CALL REWR(K1+1,1,IRR(5),K2)
514.000         IST=IST+3
515.000         GO TO 451
516.000 278     K2=0
517.000         GO TO 452
518.000 234     K=1
```

```
519.000 453   CALL REWR(IPC,1,K1,0)
520.000       K1=(K1/16)*2+1
521.000       CALL REWR(IRR(K1)*256+IRR(K1+1),1,IRR(8),K)
522.000       IST=IST+3
523.000       GO TO 244
524.000 235   K=0
525.000       GO TO 453
526.000 236   K=1
527.000 456   CALL REGRR(IPC,K2)
528.000       IF(K2.EQ.7)GO TO 454
529.000       K1=IRR(K2)*256+IRR(K2+1)+K
530.000       IF(K1.GE.65536)K1=K1-65536
531.000       IF(K1.LT.0)K1=K1+65536
532.000       IRR(K2)=K1/256
533.000       IRR(K2+1)=K1-IRR(K2)*256
534.000 455   IST=IST+1
535.000       GO TO 244
536.000 454   ISP=ISP+K
537.000       IF(ISP.GE.65536)ISP=ISP-65536
538.000       IF(ISP.LT.0)ISP=ISP+65536
539.000       GO TO 455
540.000 237   K=-1
541.000       GO TO 456
542.000 233   CALL REGRR(IPC,K)
543.000       IF(K.EQ.7)GO TO 457
544.000       K1=IRR(K)*256+IRR(K+1)
545.000 458   K1=K1+IRR(5)*256+IRR(6)
546.000       CALL CAR(ICAR,K1,65536)
547.000       IRR(5)=K1/256
548.000       IRR(6)=K1-IRR(5)*256
549.000       IST=IST+6
550.000       GO TO 244
551.000 457   K1=ISP
552.000       GO TO 458
553.000 274   CALL REWR(IPC+1,1,K1,0)
554.000       CALL REWR(IPC+2,1,K2,0)
555.000       CALL REGRR(IPC,K)
556.000       IF(K.EQ.7) GOTO 459
557.000       IRR(K)=K2
558.000       IRR(K+1)=K1
559.000 460   IST=IST+10
560.000       IPC=IPC+3
561.000       GO TO 410
562.000 459   ISP=K2*256+K1
563.000       GO TO 460
564.000 227   CALL REGRR(IPC,K)
565.000       IF(K.EQ.7)GO TO 461
566.000       CALL REWR(ISP-1,1,IRR(K),1)
567.000       CALL REWR(ISP-2,1,IRR(K+1),1)
568.000 462   ISP=ISP-2
569.000       IST=IST+7
```

```
570.000         GO TO 244
571.000 461     CALL REWR(ISP-1,1,IRR(8),1)
572.000         K=ICAR+4*IPAR+2+16*ICY1+64*IZER+128*ISIG
573.000         CALL REWR(ISP-2,1,K,1)
574.000         GO TO 462
575.000 228     CALL REGRR(IPC,K)
576.000         IF(K.EQ.7) GO TO 463
577.000         CALL REWR(ISP+1,1,IRR(K),0)
578.000         CALL REWR(ISP,1,IRR(K+1),0)
579.000 464     ISP=ISP+2
580.000         IST=IST+6
581.000         GO TO 244
582.000 463     CALL REWR(ISP+1,1,IRR(8),0)
583.000         CALL REWR(ISP,1,K,0)
584.000         ICAR=K-(K/2)*2
585.000          K=K/4
586.000         IPAR=K-(K/2)*2
587.000         K=K/4
588.000         ICY1=K-(K/2)*2
589.000         K=K/4
590.000         IZER=K-(K/2)*2
591.000         K=K/2
592.000         ISIG=K-(K/2)*2
593.000         GO TO 464
594.000 254     CALL REWR(IPC+1,1,K,0)
595.000         PRINT 465,JJ,IPC,K
596.000 465     FORMAT('AT LINE NO ',I5,' (PC= ',I4,' ) INPUT NO ',I4,' IS:')
597.000         READ 466,K1
598.000 466     FORMAT(I)
599.000         IRR(8)=K1
600.000         IPC=IPC+2
601.000         IST=IST+10
602.000         GO TO 410
603.000 216     IF(IEI.EQ.0)GO TO 485
604.000         PRINT 468,JJ
605.000 468     FORMAT('HALT AT LINE NO ',I5,' RESTART INST.(0-7):')
606.000 471     READ 466,K1
607.000         IEI=0
608.000         K3=K1*8
609.000         IST=IST+7
610.000         GO TO 501
611.000 279     IF(IEI.EQ.0) GO TO 410
612.000         PRINT 470,JJ
613.000 470     FORMAT('AT LINE NO ',I5,' RESTART INST.(0-7):')
614.000         GO TO 471
615.000 280     PRINT 472,JJ,IPC,IRR(8),(IRR(K),K=1,6),ISP,IST,ICAR,IZER,
616.000        1ISIG,IPAR,IEI
617.000 472     FORMAT(' LINE    PC      A       B       C       D       E'
618.000        1'       H       L      SP   STEP    CAR    ZER    SIG    PAR'
619.000        2'    EI',/,16I7)
620.000         GO TO 410
```

```
621.000 81     JJ=0
622.000        IPC=0
623.000 415    JJ=JJ+1
624.000        CALL SI(JJ,1,LABP,LAB,INST,IR1P,IR2P,IR12,I)
625.000        IF(INST.GE.56.AND.INST.LT.74) CALL LABEL(I,IR12(1),
625.100       1IR12(2),LABR,IPCR,IPC,LABC)
625.200        IF(INST.GE.56.AND.INST.LT.74)GOTO 415
626.000        IF(INST.EQ.81)GO TO 412
627.000        IF(INST.EQ.85)GO TO 504
627.100        IF(INST.EQ.86)GOTO 505
628.000        IF (INST.GE.79)GO TO 415
629.000 430    IF(INST.GE.56)IPC=IPC+1
630.000        IF(INST.GE.45)IPC=IPC+1
631.000        IPC=IPC+1
632.000        GO TO 415
632.100 505    CALL CTE(I,IR12(1),IR12(2),K1)
632.200        K1=K1+1
632.300        CALL LABEL(I,IR12(3),IR12(4),LABR,IPCR,K1,LABC)
632.400        GOTO 415
633.000 412    PRINT474
634.000 474    FORMAT('END OF ASSEMBLING')
635.000        IF(LABC.EQ.0)GOTO 485
636.000        PRINT 478
637.000   478  FORMAT(//,'THE LABELS ARE, WITH THEIR ADDRESSES')
638.000        DO 477 J=1,LABC
639.000   477  PRINT 479,IPCR(J),LABR(J)
640.000   479  FORMAT(5X,I8,5X,A4)
641.000        WRITE(2'NC(1))(IBUF(1,J),J=1,256)
642.000 485    JJ=0
643.000        PRINT 503
644.000 503    FORMAT('DO YOU WANT A SIMULATION? YES=1,NO=0')
645.000        READ 466,K3
645.100        IST=0
646.000        IF(K3.NE.1) GO TO 281
647.000        IPC=0
648.000 410    JJ=JJ+1
649.000        CALL SI(JJ,2,LABP,LAB,INST,IR1P,IR2P,IR12,I)
650.000        GOTO(201,202,203,204,205,206,207,208,209,210,211,212,
651.000       1213,214,215,216,217,218,219,220,221,222,223,224,225,
652.000       2226,227,228,229,230,231,232,233,234,235,236,237,238,
653.000       3239,240,241,242,243,244,245,246,247,248,249,250,251,
654.000       4252,253,254,255,256,257,258,259,260,261,262,263,264,
655.000       5265,266,267,268,269,270,271,272,273,274,275,275,275,
656.000       6275,279,280,485,410,283,410,285),INST
657.000        GO TO 410
658.000    82  CALL CTE(I,IR12(1),IR12(2),K)
659.000        CALL CTE(I,IR12(3),IR12(4),K1)
660.000        CALL REWR(K,1,K1,1)
661.000        GOTO 409
662.000 85     CALL CTE(I,IR12(1),IR12(2),IPC)
663.000        IL=IPC+1
```

```
664.000          GO TO 409
665.000 504      CALL CTE(I,IR12(1),IR12(2),IPC)
666.000          GOTO 415
667.000 285      CALL REWR(JJ,2,IPC,0)
668.000          GO TO 410
669.000 283      CALL CTE(I,IR12(1),IR12(2),K)
670.000          IF(K+1.EQ.NC(1)) GO TO 499
671.000          READ(2'K+1)(KS(J),J=1,256)
672.000 498      PRINT 500,JJ,K,KS
673.000          GO TO 410
674.000 499      DO 497 J=1,256
675.000 497      KS(J)=IBUF(1,J)
676.000          GO TO 498
677.000 500      FORMAT('AT LINE NO ',I5,' MEMORY MAP NO ',I4,' IS:',/,16(16I6,/))
678.000 84       CALL CTE(I,IR12(1),IR12(2),K)
679.000          IPCR(LABC)=K
680.000          GOTO 409
681.000 281      WRITE(2'NC(1))(IBUF(1,J),J=1,256)
682.000 467      CALL EXIT
683.000          END
684.000          SUBROUTINE REG(I,LB,LF,J)
685.000          DIMENSION KR(10),I(32)
686.000          DATA KR/4HB    ,4HC    ,4HD    ,4HE    ,4HH    ,4HL    ,
687.000         14HM    ,4HA    ,4HSP   ,4HPSW /
688.000          DECODE(LF,2,I)LB-1,IR
689.000        2 FORMAT(NX,A4)
690.000           DO 1 J=1,10
691.000        1 IF(IR.EQ.KR(J)) RETURN
692.000          PRINT 480,JJ
693.000 480      FORMAT('WRONG ARGUMENTS AT LINE ',I5)
694.000          RETURN
695.000          END
696.000          SUBROUTINE CTE(I,LB,LF,IR)
697.000          DIMENSION I(32)
698.000           DECODE(LF,2,I)LB-1,LF-LB+1,IR
699.000        2 FORMAT(NX,IN)
700.000          RETURN
701.000          END
702.000          SUBROUTINE REWR (IAD,NF,K,M)
703.000          COMMON IBUF(3,256),NC(3),IFL(3)
704.000          K1=IAD/256
705.000          K2=IAD-K1*256+1
706.000          K1=K1+1
707.000          IF(K1.NE.NC(NF)) GO TO 1
708.000        6 IF (M) 2,3,2
709.000 3        K=IBUF(NF,K2)
710.000          RETURN
711.000 2        IBUF(NF,K2)=K
712.000          IFL(NF)=1
713.000          RETURN
714.000 1        IF(IFL(NF)) 4,5,4
```

```
715.000 5      READ(NF+1'K1)(IBUF(NF,J),J=1,256)
716.000        NC(NF)=K1
717.000        GO TO 6
718.000 4      WRITE(NF+1'NC(NF))(IBUF(NF,J),J=1,256)
719.000        IFL(NF)=0
720.000        GO TO 5
721.000        END
722.000        SUBROUTINE WRI(I,J,IPC)
723.000        CALL REWR(IPC,1,I,1)
724.000        CALL REWR(IPC+1,1,J,1)
725.000         IPC=IPC+2
726.000        RETURN
727.000        END
728.000        SUBROUTINE WR1(I,IPC)
729.000        CALL REWR(IPC,1,I,1)
730.000        IPC=IPC+1
731.000         RETURN
732.000        END
733.000        SUBROUTINE CAR(IC,I,IM)
734.000        IC=0
735.000        IF(I.LT.IM) GO TO 1
736.000         IC=1
737.000        I=I-IM
738.000        RETURN
739.000 1      IF(I.GE.0) RETURN
740.000        IC=1
741.000        I=I+IM
742.000        RETURN
743.000        END
744.000        SUBROUTINE REGRR(IPC,K)
745.000        CALL REWR(IPC,1,K1,0)
746.000        K2=K1/16
747.000        K1=K1/64
748.000        K=K2-K1*4
749.000        K=K*2+1
750.000        RETURN
751.000        END
752.000        SUBROUTINE REGR(IPC,IRR,K)
753.000        DIMENSION IRR(10)
754.000        CALL REWR(IPC,1,K,0)
755.000        K=K-(K/8)*8
756.000        IF(K.EQ.6) GO TO 1
757.000        K=IRR(K+1)
758.000        RETURN
759.000 1      CALL REWR(IRR(5)*256+IRR(6),1,K,0)
760.000        RETURN
761.000        END
762.000        SUBROUTINE ICY(ICY1,K,I)
763.000        K1=K-(K/16)*16
764.000        K2=I-(K/16)*16
765.000        ICY1=0
```

```
766.000         IF((K1+2).GE.16) ICY1=1
767.000         RETURN
768.000         END
769.000         SUBROUTINE FFZSP(IZER,ISIG,IPAR,L)
770.000         IZER=0
771.000         IF(L.EQ.0) IZER=1
772.000         ISIG=0
773.000         IF(L.GE.128) ISIG=1
774.000         IPAR=0
775.000         I=L
776.000         DO 1 J=1,8
777.000         IPAR=I-(I/2)*2+IPAR
778.000 1       I=I/2
779.000         IPAR=IABS(IPAR-(IPAR/2)*2-1)
780.000         RETURN
781.000         END
782.000         SUBROUTINE S1(I,N,L)
783.000         DIMENSION I(32)
784.000         DATA MASK/8Z000000FF/
785.000         M=1+(N-1)/4
786.000         K=N-(M-1)*4
787.000         II=ISL(I(M),8*(K-4))
788.000         L=IAND(MASK,II)
789.000         RETURN
790.000         END
791.000         SUBROUTINE SEARCH(I,NC,LB,LF,CP,PC)
792.000         INTEGER CP,PC
793.000         DIMENSION I(32)
794.000         DO 1 N=LB,LF
795.000         CALL S1(I,N,L)
796.000         IF(L.EQ.NC) GO TO 2
797.000       1 CONTINUE
798.000         CP=0
799.000         RETURN
800.000       2 CP=1
801.000         PC=N
802.000         RETURN
803.000         END
804.000         SUBROUTINE SI(J,K,LABP,LAB,INST,IR1P,IR2P,IR12,I)
805.000         DIMENSION INS(86),IR12(4),I(32)
806.000         DATA INS/4HMOV ,4HINR ,4HDCR ,4HADD ,4HADC ,4HSUB ,4HSBB ,
807.000        14HANA ,4HXRA ,4HORA ,4HCMP ,4HRLC ,4HRRC ,4HRAL ,4HRAR ,
808.000        24HHLT ,4HRET ,4HRC  ,4HRNC ,4HRZ  ,4HRNZ ,4HRP  ,4HRM  ,
809.000        34HRPE ,4HRPO ,4HRST ,4HPUSH,4HPOP ,4HXCHG,4HXTHL,4HSPHL,
810.000        44HPCHL,4HDAD ,4HSTAX,4HLDAX,4HINX ,4HDCX ,4HCMA ,4HSTC ,
811.000        54HCMC ,4HDAA ,4HEI  ,4HDI  ,4HNOP ,4HMVI ,4HADI ,4HACI ,
812.000        64HSUI ,4HSBI ,4HANI ,4HXRI ,4HORI ,4HCPI ,4HIN  ,4HOUT ,
813.000        74HJMP ,4HJC  ,4HJNC ,4HJZ  ,4HJNZ ,4HJP  ,4HJM  ,4HJPE ,
814.000        84HJPO ,4HCALL,4HCC  ,4HCNC ,4HCZ  ,4HCNZ ,4HCP  ,4HCM  ,
815.000        94HCPE ,4HCPO ,4HLXI ,4HSTA ,4HLDA ,4HSHLD,4HLHLD,4HINTE,
816.000        A4HPRIN,4HEND ,4HASSI,4HMAP ,4HVAR ,4HBASE,4HASSD/
```

```
817.000         INTEGER CP,PC
818.000         ENCODE (128,12,I)
819.000      12 FORMAT(4H      )
820.000         CALL DIRECT READ(1,I,32,J)
821.000         IR1P=0
822.000         IR2P=0
823.000         LB=1
824.000         LF=128
825.000         CALL SEARCH(I,94,LB,LF,CP,PC)
826.000         IF(CP.EQ.1) LF=PC-1
827.000         CALL CLSP(I,LF,-1)
828.000         LABP=0
829.000         CALL CLSP(I,LB,1)
830.000         CALL SEARCH(I,122,LB,LF,CP,PC)
831.000         IF(CP.EQ.1) GO TO 2
832.000 13      CALL CLSP(I,LB,1)
833.000         CALL SEARCH(I,64,LB,LF,CP,PC)
834.000         LFF=LF
835.000         IF(CP.EQ.1) LFF=PC-1
836.000         GO TO 4
837.000      2  IF(K.NE.0) GO TO 6
838.000         LABP=1
839.000         LAB=4H
840.000         LK=1
841.000         DO 7 L1=LB,PC-1
842.000         CALL S1(I,L1,L)
843.000         IF(L.EQ.1H )GO TO 7
844.000         L=ISL(L,8*(4-LK))
845.000         MASK=ISC(8ZFFFFFF00,8*(4-LK))
846.000         LAB=IOR(IAND(MASK,LAB),L)
847.000         IF(LK.EQ.4) GO TO 6
848.000 7       LK=LK+1
851.000      6  LB=PC+1
852.000         GO TO 13
853.000      4  INST=4H
854.000         LK=1
855.000         DO 9 L1=LB,LB+3
856.000         CALL S1(I,L1,L)
857.000         IF(L1.GT.LFF)L=64
858.000         L=ISL(L,8*(4-LK))
859.000         MASK=ISC(8ZFFFFFF00,8*(4-LK))
860.000         INST=IOR(IAND(MASK,INST),L)
861.000 9       LK=LK+1
863.000 14      LB=LFF+1
864.000         DO 10 L1=1,86
865.000         IF(INS(L1).EQ.INST) GO TO 11
866.000     10  CONTINUE
867.000     11  INST=L1
869.000         CALL CLSP(I,LF,-1)
870.000         IF(LB.GT.LF) RETURN
871.000         IR1P=1
```

```
 872.000         CALL CLSP(I,LB,1)
 873.000         CALL SEARCH(I,107,LB,LF,CP,PC)
 874.000         IR12(1)=LB
 875.000         IF(CP.EQ.1) GO TO 1
 876.000         IR12(2)=LF
 877.000         RETURN
 878.000       1 LK=PC-1
 879.000         CALL CLSP(I,LK,-1)
 880.000         IR12(2)=LK
 881.000         IR2P=1
 882.000         LB=PC+1
 883.000         CALL CLSP(I,LB,1)
 884.000         IR12(3)=LB
 885.000         IR12(4)=LF
 886.000         RETURN
 887.000         END
 888.000         SUBROUTINE CLSP(I,LB,J)
 889.000         DIMENSION I(32)
 890.000       2 CALL S1(I,LB,L)
 891.000         IF(L.NE.64)RETURN
 892.000         LB=LB+J
 893.000         GOTO 2
 894.000         END
 895.000         SUBROUTINE LABEL(I,K1,K2,LABR,IPCR,K3,LABC)
 896.000         DIMENSIONI(32),LABR(200),IPCR(200)
 897.000         DECODE (K2,1,I)K1-1,K
 898.000 1       FORMAT(NX,A4)
 899.000         LAI=0
 900.000         DO 2 J=1,LABC
 901.000         IF(K.EQ.LABR(J)) LAI=LAI+1
 902.000 2       IF(K.EQ.LABR(J)) LAB=IPCR(J)
 903.000         IF(LAI.EQ.1)GOTO 3
 904.000         PRINT 4,LAI,K
 905.000 4       FORMAT('THERE ARE ',I3,' LABELS CALLED ',A4)
 906.000 3       K3=K3+1
 907.000         K1=LAB/256
 908.000         LAB=LAB-K1*256
 909.000         CALL WRI(LAB,K1,K3)
 910.000         RETURN
 911.000         END
--EOF HIT AFTER  911.
*END
```

APPENDIX B

Listing of MORP

```
 EDIT MOR
*TY 1=25
   1.000          INTEGER TYPE(2),SIZE(2)
   2.000          INTEGER R(256,4)
   3.000          TYPE(1)=2
   4.000          TYPE(2)=2
   5.000          SIZE(1)=1
   6.000          SIZE(2)=1024
   7.000          PRINT 2
   8.000 2        FORMAT('WHAT IS THE STARTING MEMORY PAGE?')
   9.000          DEFINE FILE 2(256,256,U,ICA)
  10.000          READ 1,N
  11.000       1  FORMAT(I)
  12.000         N=N+1
  13.000          READ(2'N)(R(I,1),I=1,256)
  14.000          READ(2'N+1)(R(I,2),I=1,256)
  15.000          READ(2'N+2)(R(I,3),I=1,256)
  16.000          READ(2'N+3)(R(I,4),I=1,256)
  17.000          CALL FTIE(5,'FA')
  18.000          CALL FREPLACE(5,1,R,SIZE,TYPE)
  19.000          CALL FUNTIE(5)
  20.000          CALL EXIT
  21.000           END
==EOF HIT AFTER    21.
*END
```

APPENDIX C

Listing of ASSPROM

```
      ∇ASSPROM[□]∇
      ∇ASSPROM
[1]   'FA'FTIE 6
[2]   A←FREAD 6,1
[3]   FUNTIE 6
[4]   'D''ONT FORGET TO SAVE'
      ∇
```

CRDV R-4124/78 (NON CLASSIFIE)

Bureau - Recherche et Développement, MDN, Canada.
CRDV, C.P. 880, Courcelette, Qué. GOA 1RO

"Version améliorée d'un assembleur et d'un simulateur pour le microprocesseur 8080" (NC)
par J.N. Bérubé, R. Carbonneau, B. Montminy, P. Côté

Nous présentons ici une version améliorée d'un programme servant à la traduction des mnémoniques employés dans la programmation du microprocesseur 8080 en code machine. Ce programme, écrit en langage FORTRAN, permet également la simulation exacte du comportement du microprocesseur dans des applications réelles de même que l'emploi d'étiquettes pour les instructions de branchement. Il permet en outre la simulation d'interruptions et l'impression des résultats intermédiaires. Cette nouvelle version, en plus d'être dix fois plus rapide que la version en APL, peut être utilisée sur tout ordinateur possédant un compilateur FORTRAN. (NC)

CRDV R-4124/78 (NON CLASSIFIE)

Bureau - Recherche et Développement, MDN, Canada.
CRDV, C.P. 880, Courcelette, Qué. GOA 1RO

"Version améliorée d'un assembleur et d'un simulateur pour le microprocesseur 8080" (NC)
par J.N. Bérubé, R. Carbonneau, B. Montminy, P. Côté

Nous présentons ici une version améliorée d'un programme servant à la traduction des mnémoniques employés dans la programmation du microprocesseur 8080 en code machine. Ce programme, écrit en langage FORTRAN, permet également la simulation exacte du comportement du microprocesseur dans des applications réelles de même que l'emploi d'étiquettes pour les instructions de branchement. Il permet en outre la simulation d'interruptions et l'impression des résultats intermédiaires. Cette nouvelle version, en plus d'être dix fois plus rapide que la version en APL, peut être utilisée sur tout ordinateur possédant un compilateur FORTRAN. (NC)

CRDV R-4124/78 (NON CLASSIFIE)

Bureau - Recherche et Développement, MDN, Canada.
CRDV, C.P. 880, Courcelette, Qué. GOA 1RO

"Version améliorée d'un assembleur et d'un simulateur pour le microprocesseur 8080" (NC)
par J.N. Bérubé, R. Carbonneau, B. Montminy, P. Côté

Nous présentons ici une version améliorée d'un programme servant à la traduction des mnémoniques employés dans la programmation du microprocesseur 8080 en code machine. Ce programme, écrit en langage FORTRAN, permet également la simulation exacte du comportement du microprocesseur dans des applications réelles de même que l'emploi d'étiquettes pour les instructions de branchement. Il permet en outre la simulation d'interruptions et l'impression des résultats intermédiaires. Cette nouvelle version, en plus d'être dix fois plus rapide que la version en APL, peut être utilisée sur tout ordinateur possédant un compilateur FORTRAN. (NC)

CRDV R-4124/78 (NON CLASSIFIE)

Bureau - Recherche et Développement, MDN, Canada.
CRDV, C.P. 880, Courcelette, Qué. GOA 1RO

"Version améliorée d'un assembleur et d'un simulateur pour le microprocesseur 8080" (NC)
par J.N. Bérubé, R. Carbonneau, B. Montminy, P. Côté

Nous présentons ici une version améliorée d'un programme servant à la traduction des mnémoniques employés dans la programmation du microprocesseur 8080 en code machine. Ce programme, écrit en langage FORTRAN, permet également la simulation exacte du comportement du microprocesseur dans des applications réelles de même que l'emploi d'étiquettes pour les instructions de branchement. Il permet en outre la simulation d'interruptions et l'impression des résultats intermédiaires. Cette nouvelle version, en plus d'être dix fois plus rapide que la version en APL, peut être utilisée sur tout ordinateur possédant un compilateur FORTRAN. (NC)

DREV R-4124/78 (UNCLASSIFIED)

Research and Development Branch, DND, Canada.
DREV, P.O. Box 880, Courcelette, Qué. G0A 1R0

"An Improved Version of an Assembler and Simulator for the 8080 Microprocessor" (U)
by J.N. Bérubé, R. Carbonneau, B. Montminy, P. Côté

This report describes an improved version of an assembler and simulator being used to translate man-readable statements into machine-understandable code. This program, written in FORTRAN, allows programming of the 8080 microprocessor in symbolic language as well as the use of labels for jump instructions while the simulator duplicates exactly the behavior of the microprocessor in real applications. It is also possible to simulate interrupts and print out intermediate results. This new version may be used on any computer with a FORTRAN compiler while the former APL version was slower by a factor of ten and limited to a computer with an APL interpreter. (U)

DREV R-4124/78 (UNCLASSIFIED)

Research and Development Branch, DND, Canada.
DREV, P.O. Box 880, Courcelette, Qué. G0A 1R0

"An Improved Version of an Assembler and Simulator for the 8080 Microprocessor" (U)
by J.N. Bérubé, R. Carbonneau, B. Montminy, P. Côté

This report describes an improved version of an assembler and simulator being used to translate man-readable statements into machine-understandable code. This program, written in FORTRAN, allows programming of the 8080 microprocessor in symbolic language as well as the use of labels for jump instructions while the simulator duplicates exactly the behavior of the microprocessor in real applications. It is also possible to simulate interrupts and print out intermediate results. This new version may be used on any computer with a FORTRAN compiler while the former APL version was slower by a factor of ten and limited to a computer with an APL interpreter. (U)

DREV R-4124/78 (UNCLASSIFIED)

Research and Development Branch, DND, Canada.
DREV, P.O. Box 880, Courcelette, Qué. G0A 1R0

"An Improved Version of an Assembler and Simulator for the 8080 Microprocessor" (U)
by J.N. Bérubé, R. Carbonneau, B. Montminy, P. Côté

This report describes an improved version of an assembler and simulator being used to translate man-readable statements into machine-understandable code. This program, written in FORTRAN, allows programming of the 8080 microprocessor in symbolic language as well as the use of labels for jump instructions while the simulator duplicates exactly the behavior of the microprocessor in real applications. It is also possible to simulate interrupts and print out intermediate results. This new version may be used on any computer with a FORTRAN compiler while the former APL version was slower by a factor of ten and limited to a computer with an APL interpreter. (U)

DREV R-4124/78 (UNCLASSIFIED)

Research and Development Branch, DND, Canada.
DREV, P.O. Box 880, Courcelette, Qué. G0A 1R0

"An Improved Version of an Assembler and Simulator for the 8080 Microprocessor" (U)
by J.N. Bérubé, R. Carbonneau, B. Montminy, P. Côté

This report describes an improved version of an assembler and simulator being used to translate man-readable statements into machine-understandable code. This program, written in FORTRAN, allows programming of the 8080 microprocessor in symbolic language as well as the use of labels for jump instructions while the simulator duplicates exactly the behavior of the microprocessor in real applications. It is also possible to simulate interrupts and print out intermediate results. This new version may be used on any computer with a FORTRAN compiler while the former APL version was slower by a factor of ten and limited to a computer with an APL interpreter. (U)